

## LA-UR-14-28975

Approved for public release; distribution is unlimited.

Title: User Guide for the LSP-Slice PIC code

Author(s): Ekdahl, Carl August Jr.

Intended for: Report

Issued: 2014-11-19

---

**Disclaimer:**

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

# User Guide for the LSP-Slice PIC code

Carl Ekdahl

## CONTENTS

- I. INTRODUCTION
- II. SETTING UP LSP
- III. RUNNING THE CODE
- IV. EXAMPLE

## I. INTRODUCTION

LSP-Slice is a particle in cell (PIC) code authored by Carsten Thoma and Thomas Hughes at Voss Scientific, LLC [1]. It is based on the Large Scale Plasma (LSP) code, and was developed to solve problems related to the transport of intense relativistic electron beams (IREBs) through linear induction accelerators (LIAs). In particular, it has been extensively used to investigate emittance growth in the LIAs of the Dual-Axis Radiography of Hydrodynamic Tests (DARHT) facility at Los Alamos [2] [3], and to predict beam transport and dynamics in new LIA designs [4]

Essentially, the code follows a transverse slice of the distribution of beam electrons as it propagates through the LIA while being accelerated by the electric field of the cells, and focused by the magnetic field of the solenoids. By limiting the calculations to a slice of the beam, rather than the entire beam pulse, the calculation usually takes only minutes to hours, rather than weeks to months. Two versions of the code exist, “1D” and “2D.” The dimensionality refers to the transvers spatial coordinate; the 1D version is axially symmetric, whereas the 2D version is Cartesian, permitting offset beam injection and transport, and realistic visualization of particle distributions.

The code is based on the Large-Scale Plasma (LSP) code, and is executed through the LSP user interface. In what follows, setting LSP to run LSP-Slice is described, as well as how to run the code, and a few examples are provided. Also discussed are the physics models underlying the code, and some of the physics pitfalls that may surprise the unwary user.

## II. SETTING UP LSP

The following instructions are specific to setting up an HP Z820 32-processor workstation running the 64-bit Microsoft Windows 7 Pro operating system (OS). On this computer, LSP V6.91 is compiled using Microsoft Visual Studio Express 2010. This compiler is freely available at:

[http://www.visualstudio.com/downloads/download-visual-studio-vs#DownloadFamilies\\_4](http://www.visualstudio.com/downloads/download-visual-studio-vs#DownloadFamilies_4)

LSP can be run with either a single or multiple processors. In the DARHT Physics group it is run on Windows OS computers with 4 to 32 processors. These instructions are geared toward multiple-processor runs of LSP-Slice. To facilitate multiple-processor runs, the Message Passing Interface MPICH2 V1.2.1 or newer is required. MPICH 2 1.2.1 or later (x64 binary) is available at:

<http://www.mcs.anl.gov/research/projects/mpich2/>

or:

[http://www.vosssci.com/upload/ATK\\_Lsp/Common/VS-MP/](http://www.vosssci.com/upload/ATK_Lsp/Common/VS-MP/)

More information about these utilities, their installation, and testing, can be found in Appendix A.

Open the LSP GUI and load an input deck to compile. Navigate to <tools>-<LSP>-<compile> and make sure that the MPI location is correct.

### III. RUNNING THE CODE

For nominal accelerator sizes, single processor runs are painfully slow, greatly exceeding the tolerance of some physicists. On a 32-processor Windows machine, 1D simulations of DARHT usually take a few minutes, and are used to survey a range of initial conditions, or LIA configurations. The 2D simulations typically take a few hours, and have been reserved for only the most interesting cases uncovered in the 1D explorations. In all comparisons to date, the agreement between the 1D and 2D simulations has been excellent.

There are several files required for a multi-processor LSP-Slice computation:

- External magnetic field on axis;  $B_z(0,0,z)$ , e.g. Bz.dat (an ASCII file)
- External electric field on axis,  $E_z(0,0,z)$ , e.g. Ez.dat (an ASCII file)
- An initial particle distribution created by extractor.exe; e. g., pext12.p4 (a p4 file)
- An LSP input script for running the first 10 steps, e. g. sim1D\_start.lsp (an ASCII file)
- An LSP input script for restarting LSP and completing the run, e. g. sim1D.lsp (an ASCII file)

Running LSP-Slice is a three step process. One first runs extractor.exe to extract a particle distribution slice from a full LSP run. Next, one runs slice with one processor to generate the magnetic field with the magnetostatic solve, which does not work in a multiprocessor mode. This step dumps a restart.p4 file from which the multiprocess run can be restarted. Finally, one runs slice with multiprocessor, restarting from the single processor dump.

#### A. Running EXTRACTOR

The initial particle distribution is generated by the extractor.exe code. If you don't have Cygwin installed on your computer, you must have Cygwin.dll in the same directory as extractor.exe. When executed, extractor opens a DOS CMD window in which it prompts for the following user inputs:

1. Transverse dimensionality, 1D or 2D (user enters 0 or 1)
2. Beam Current in kA
3. Wall Potential in MeV (This is what the beam electron KE would be if not for space-charge depression)
4. Wall radius in cm (This has to be constant throughout the accelerator)
5. Beam edge radius in cm (This is the envelope radius =  $\sqrt{2}$ \*rms radius)
6. Beam "Angle" at edge in mr (This is the convergence of the beam envelope. Early experience showed that the best agreement with full PIC simulations was achieved when

the launch point was at a point where the beam envelope is at an extremum, and this angle is then zero)

7. Axial slice position in cm. (In same coordinates as the Bz and Ez files)
8. Beam emittance in cm-radian (This is normalized emittance =  $\beta\gamma \times 4 \text{ rms}$ )
9. Bz(0,0,z) at launch position in Gauss.
10. Then,
  - a. **for a 1D run:** Number of particles in slice. (4000 generally gives good results for DARHT, but it should be varied to test convergence whenever a radically different accelerator architecture is first tried, or if a wildly unmatched beam results from the choice of initial conditions. )
  - b. **for a 2D run:** Initial offset and angle of beam
  - c. **for a 2D run:** number of cells and number of particles/cell (50 cells and 3 particles/cell produces 70,688 macroparticles, and generally gives good results for DARHT, but it should be varied to test convergence. )
11. Integer identifier. (This number becomes part of the file name. For example, entering 12 produces a file named pext12.p4. The \*.lsp input decks are set up to recognize 12 for a 1D run and 8 for a 2D run.)

#### B. Modifying the simND.lsp input decks

- 1) [control] Set *time\_limit\_ns* to the length of the simulation, set dumps to suite
- 2) [grid] set *z\_min* to z0 (initial position) (radial grid is large enough to accommodate both DARHT axes)
- 3) [objects] set *base* to 0 0 z0
- 4) [particle creation] set *from* to 0 0 z0; set *to* to 12.0 6.2831853 z0
- 5) [probes] set *at* or *from* to 0 0 z0, add diagnostics to suite

#### C. Running a 1D simulation

- 1) Run extractor.exe choosing 12 as the integer to name the file (if you want to use some other integer, be sure to change the file name in the [Particle Creation] section of the sim1D\_start.lsp script).
- 2) Move this pext12.p4 file into the directory containing the sim1D.lsp scripts you want to execute.

From the LSP GUI;

- 1) Compile sim1D.lsp (correct compiler options are set in the script)
- 2) Open and Run sim1D\_start.lsp. (Ignore warnings, e.g., answer “No” to MULTI\_PROCESS flag warning, and then ‘yes’ to continue anyway.) This should run 10 steps producing
- 3) Open and Run sim1D.lsp with Lsp preopt entry: mpiexec.exe -np 32 (or however many you have) and the -r box checked (for restarting from the file restart.p4)

#### C. Running a 2D simulation

- 1) Run extractor.exe choosing 8 as the integer to name the file (if you want to use some other integer, be sure to change the file name in the [Particle Creation] section of the sim2D\_start.lsp script).
- 2) Move this pext8.p4 file into the directory containing the sim1D.lsp scripts you want to execute.

From the LSP GUI;

- 1) Compile sim2D.lsp (correct compiler options are set in the script)
- 2) Open and Run sim2D\_start.lsp. (Ignore warnings, e.g., answer “No” to MULTI\_PROCESS flag warning, and then ‘yes’ to continue anyway.) This should run 10 steps producing
- 3) Open and Run sim2D.lsp with an entry in Lsp preopt box: mpiexec.exe -np 32 (or however many you have) and the -r box checked (for restarting from the file restart.p4 dumped by the first run)

#### IV. EXAMPLE

All files needed to run LSP-Slice for an example problem are included in a zipped distribution. The example problem is matched beam transport and acceleration through DARHT Axis-I using the nominal 2-inch cathode tune. Initial conditions were derived from experimental measurements just outside the diode, and extrapolated back to the LSP-Slice launch point using our XTR envelope code. The launch point is the location where the beam expansion out of the diode is at a maximum, before it is focused down with the anode solenoid. Initial parameters at the launch point are (XTR nomenclature):

1. Beam current,  $I_b = 1.75$  kA.
2. Wall Potential,  $\Phi = 3.71$  MV
3. Beam Envelope radius,  $r_0 = 4.421$
4. Beam envelope convergence angle,  $r_0 p = 0.00$  mr
5. Normalized emittance,  $\epsilon_n = 0.100$  cm-radian
6. Position,  $z_0 = 65.52$  cm
7. Magnetic field,  $B_z = 652.7$  G

Axis-I has a 7.5-cm beam pipe from the anode out to the final focus magnet. The external magnetic field was dumped from an XTR simulation, and is based on a model of solenoids fitted to experimental field maps. The external electric field is based on a TRICOMP electrostatic (ESTAT [5]) simulation of the accelerating gap region.

##### A. Timing

The 1D run with 4,000 macropoles and 0.2 cm steps takes < 3 minutes of CPU time with 32 processors. On the same computer, the 2D run with 70,668 macropoles takes 1.47 Hr.

##### B. Results

Figure 1 shows the 1D PIC envelope compared with the XTR results. The LSP-Slice PIC code calculates the rms radius, which is converted to an equivalent envelope by  $r_{env} = r_{rms} \sqrt{2}$ . Figure 2 compares 1D and 2D results.

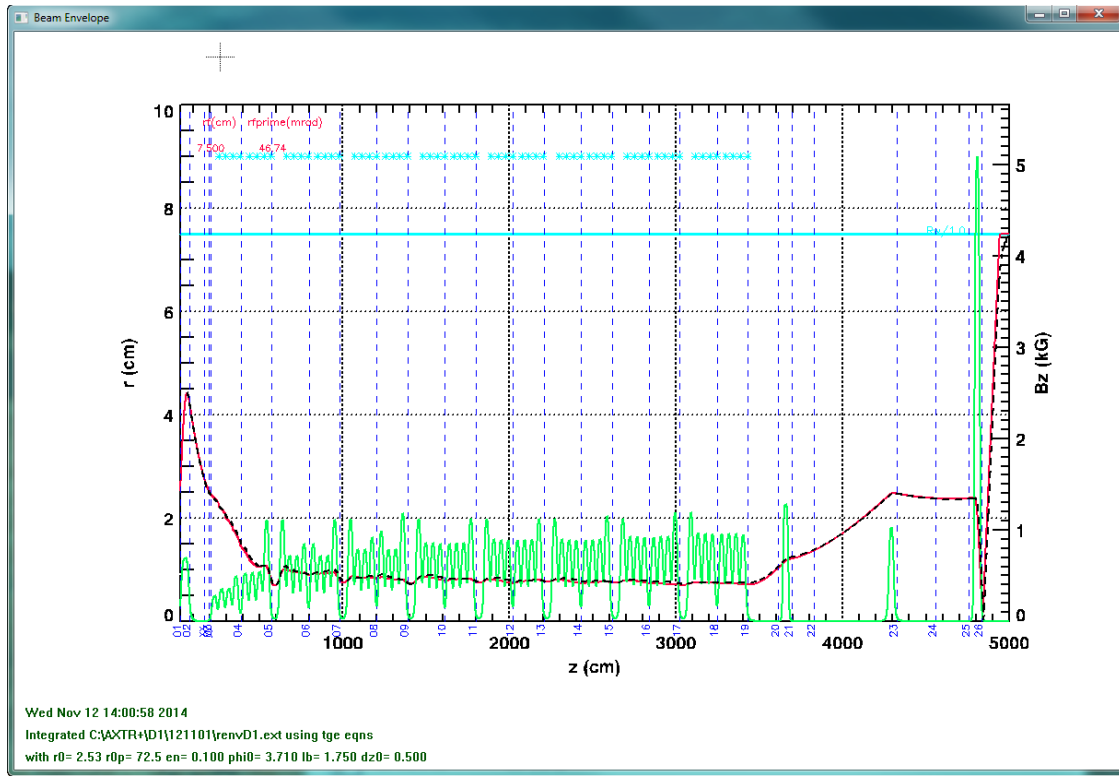


Figure 1: Comparison of XTR envelope code with 1D LSP-Slice for the nominal DARHT Axis-I tune. (red) Beam envelope calculated by XTR. (dashed black) Beam envelope from LSP-Slice PIC code. (green) Magnetic guide field on axis, right scale. (solid cyan) Beam pipe wall. (cyan asterisks) Accelerating cell potential. (blue dashed) BPM locations.

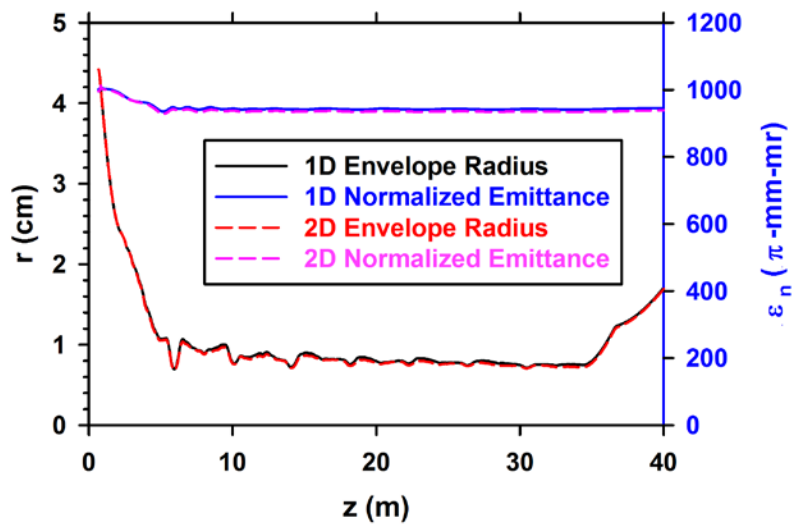


Figure 2: Comparison of 1D and 2D LSP-Slice calculations of envelope radius and normalized emittance for DARHT Axis-I.

## References

- [1] C. Thoma and T. P. Hughes, "A beam-slice algorithm for transport of the DARHT-2 accelerator," in *Part. Acc. Conf.*, 2007.
- [2] C. Ekdahl, "Tuning the DARHT long-pulse linear induction accelerator," *IEEE Trans. Plasma Sci.*, vol. 41, pp. 2774 - 2780, 2013.
- [3] C. Ekdahl and et al., "Emittance growth in linear induction accelerators," in *20th Int. Conf. High Power Part. Beams*, Washington, DC, 2014.
- [4] C. Ekdahl, "Beam Dynamics for ARIA," Los Alamos National laboratory Report LA-UR-14-27454, 2014.
- [5] S. Humphries, "Technical information: TriComp Series," Field Precision, LLC, 2013. [Online]. Available: [www.fieldp.com/technical.html](http://www.fieldp.com/technical.html).

## **This guide details how to install and configure LspSuite for the Windows 7 64bit operating system using VS Express 2010.**

### **Requirements:**

- 1) Microsoft Visual Studio Express 2010 C++

Freely available at:

[www.visualstudio.com/downloads/download-visual-studio-vs#DownloadFamilies\\_4](http://www.visualstudio.com/downloads/download-visual-studio-vs#DownloadFamilies_4)

- 2) Windows SDK for Windows Server 2008 and .NET Framework 3.5

Available at:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=F26B1AA4-741A-433A-9BE5-FA919850BDBF&displaylang=en>

- 3) VCE64BIT.zip (VS Express 64bit patch script)

Available at:

[http://www.vosssci.com/upload/ATK\\_Lsp/Common/VS-MP/VCE64BIT.zip](http://www.vosssci.com/upload/ATK_Lsp/Common/VS-MP/VCE64BIT.zip)

- 4) Python 2.6.5 (x64 binary) (recommended but optional)

Available at: <http://www.python.org/download/>

Or: [http://www.vosssci.com/upload/ATK\\_Lsp/Common/VS-MP/](http://www.vosssci.com/upload/ATK_Lsp/Common/VS-MP/)

- 5) Mpich2 1.2.1 or later (x64 binary) (required for multi-process runs)

Available at: <http://www.mcs.anl.gov/research/projects/mpich2/>

Or: [http://www.vosssci.com/upload/ATK\\_Lsp/Common/VS-MP/](http://www.vosssci.com/upload/ATK_Lsp/Common/VS-MP/)

6) LspSuite 6.85 or later

# Install Visual Studio Express 2010 C++

Scroll down to the Visual Studio 2010 Express section:

The screenshot shows the Visual Studio 2010 Express download page. The top navigation bar includes links for Products, Pricing, Explore, Downloads (active), Get Started, Resources, News, and Support. A green button for 'Free Visual Studio trial' is also present. Below the navigation bar, there are tabs for Free trials, Express, Additional software, 2010 Express (active), and Updates. The main content area is titled 'Visual Studio 2010 Express' and features a blue header for 'Visual C++ 2010 Express'. Under this header, there are two columns. The left column, titled 'Visual C++ 2010 Express', describes the product and provides a 'Download language' dropdown menu set to 'English'. Below this, there are 'Installation options' with a button to 'Install now'. The right column, titled 'Microsoft Captions Language Interface Pack (CLIP)', describes the CLIP tool and lists supported languages: Arabic, Greek, Hebrew, Hindi, Hungarian, Malay, Malayalam, Oriya, Tamil, and Thai. At the bottom of the page, there are four blue buttons for other Visual Studio 2010 Express editions: Visual C#, Visual Basic, Visual Web Developer, and Visual Studio 2010 Express All-in-One ISO.

Visual Studio

MSDN Subscriptions Sign in

Products Pricing Explore Downloads Get Started Resources News Support

Free Visual Studio trial

Free trials Express Additional software 2010 Express Updates

## Visual Studio 2010 Express

### Visual C++ 2010 Express

Build custom applications in Visual C++, a powerful language that gives deep and detailed control when building either native Windows (COM+) applications or .NET Framework-managed Windows applications. After installation, you can try this product for up to 30 days. You must register to obtain a free product key for ongoing use after 30 days.

**Download language**

English

**Installation options**

Visual C++ 2010 Express - English  
Install now

### Microsoft Captions Language Interface Pack (CLIP)

The Microsoft Captions Language Interface Pack (CLIP) uses tooltip captions to display translations for common user interface elements in the Visual Studio integrated development environment (IDE). Use CLIP as a language aid, to see translations in your own dialect, update results in your own native tongue or as a learning tool.

- Arabic - العربية
- Greek - Ελληνικά
- Hebrew - עברית
- Hindi - हिन्दी
- Hungarian - Magyar
- Malay - ملايو
- Malayalam - മലയാളം
- Oriya - ଓଡ଼ିଆ
- Tamil - தமிழ்
- Thai - ไทย

[Choose Language](#)

### Visual C# 2010 Express

### Visual Basic 2010 Express

### Visual Web Developer 2010 Express

### Visual Studio 2010 Express All-in-One ISO

## Install Windows SDK for Windows Server 2008 and .NET Framework 3.5

Execute the installer and leave the default installation options.



## **Run the 64 bit patch script**

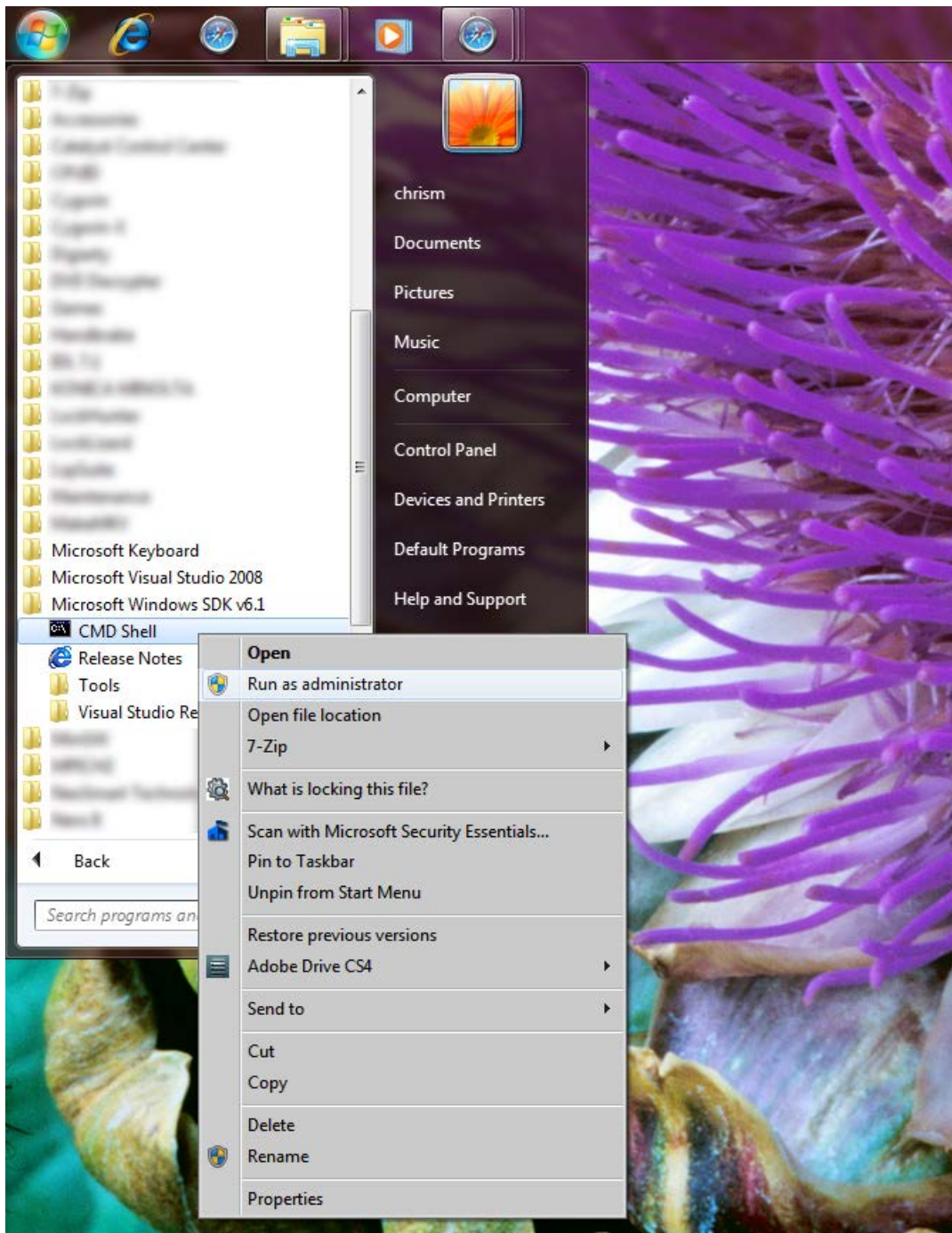
Extract the VCE64BIT.zip archive. (Right-click the file -> Extract All)

Run a command shell as administrator.

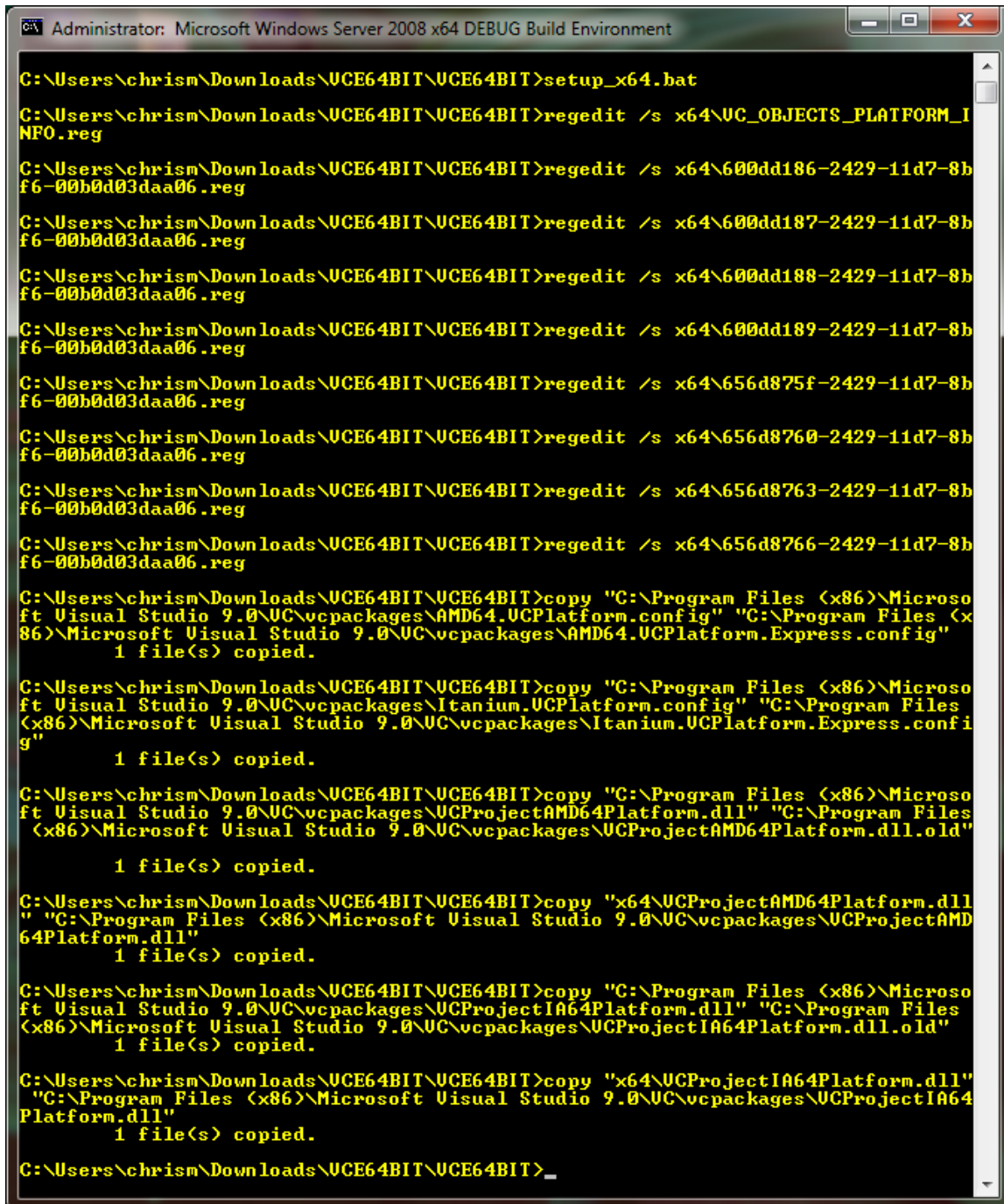
Go to: Start Menu -> All Programs -> Microsoft Windows SDK v6.1

Right Click: Cmd Shell

Select: Run as administrator



Change to the VCE64BIT directory and run setup\_x64.bat.



```
C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>setup_x64.bat

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>regedit /s x64\UC_OBJECTS_PLATFORM_I
NFO.reg

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>regedit /s x64\600dd186-2429-11d7-8b
f6-00b0d03daa06.reg

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>regedit /s x64\600dd187-2429-11d7-8b
f6-00b0d03daa06.reg

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>regedit /s x64\600dd188-2429-11d7-8b
f6-00b0d03daa06.reg

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>regedit /s x64\600dd189-2429-11d7-8b
f6-00b0d03daa06.reg

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>regedit /s x64\656d875f-2429-11d7-8b
f6-00b0d03daa06.reg

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>regedit /s x64\656d8760-2429-11d7-8b
f6-00b0d03daa06.reg

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>regedit /s x64\656d8763-2429-11d7-8b
f6-00b0d03daa06.reg

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>regedit /s x64\656d8766-2429-11d7-8b
f6-00b0d03daa06.reg

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>copy "C:\Program Files (x86)\Microso
ft Visual Studio 9.0\VC\vcpackages\AMD64.UCPlatform.config" "C:\Program Files (x
86)\Microsoft Visual Studio 9.0\VC\vcpackages\AMD64.UCPlatform.Express.config"
1 file(s) copied.

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>copy "C:\Program Files (x86)\Microso
ft Visual Studio 9.0\VC\vcpackages\Itanium.UCPlatform.config" "C:\Program Files
(x86)\Microsoft Visual Studio 9.0\VC\vcpackages\Itanium.UCPlatform.Express.conf
ig"
1 file(s) copied.

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>copy "C:\Program Files (x86)\Microso
ft Visual Studio 9.0\VC\vcpackages\UCProjectAMD64Platform.dll" "C:\Program Files
(x86)\Microsoft Visual Studio 9.0\VC\vcpackages\UCProjectAMD64Platform.dll.old"
1 file(s) copied.

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>copy "x64\UCProjectAMD64Platform.dll
" "C:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\vcpackages\UCProjectAMD
64Platform.dll"
1 file(s) copied.

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>copy "C:\Program Files (x86)\Microso
ft Visual Studio 9.0\VC\vcpackages\UCProjectIA64Platform.dll" "C:\Program Files
(x86)\Microsoft Visual Studio 9.0\VC\vcpackages\UCProjectIA64Platform.dll.old"
1 file(s) copied.

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>copy "x64\UCProjectIA64Platform.dll"
"C:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\vcpackages\UCProjectIA64
Platform.dll"
1 file(s) copied.

C:\Users\chrism\Downloads\VCE64BIT\VCE64BIT>_
```

## **Install Python**

Execute the python installer and choose the default installation options.

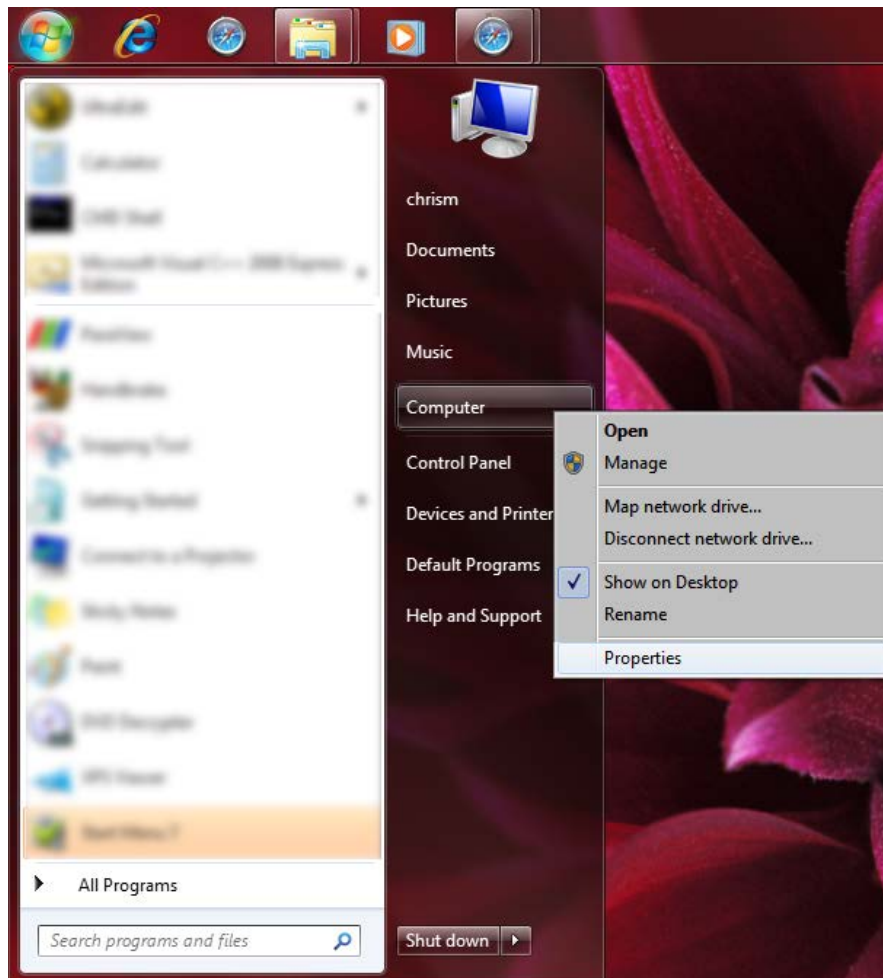
## Install MPICH2

Execute the mpich2 installer and choose the default install options.

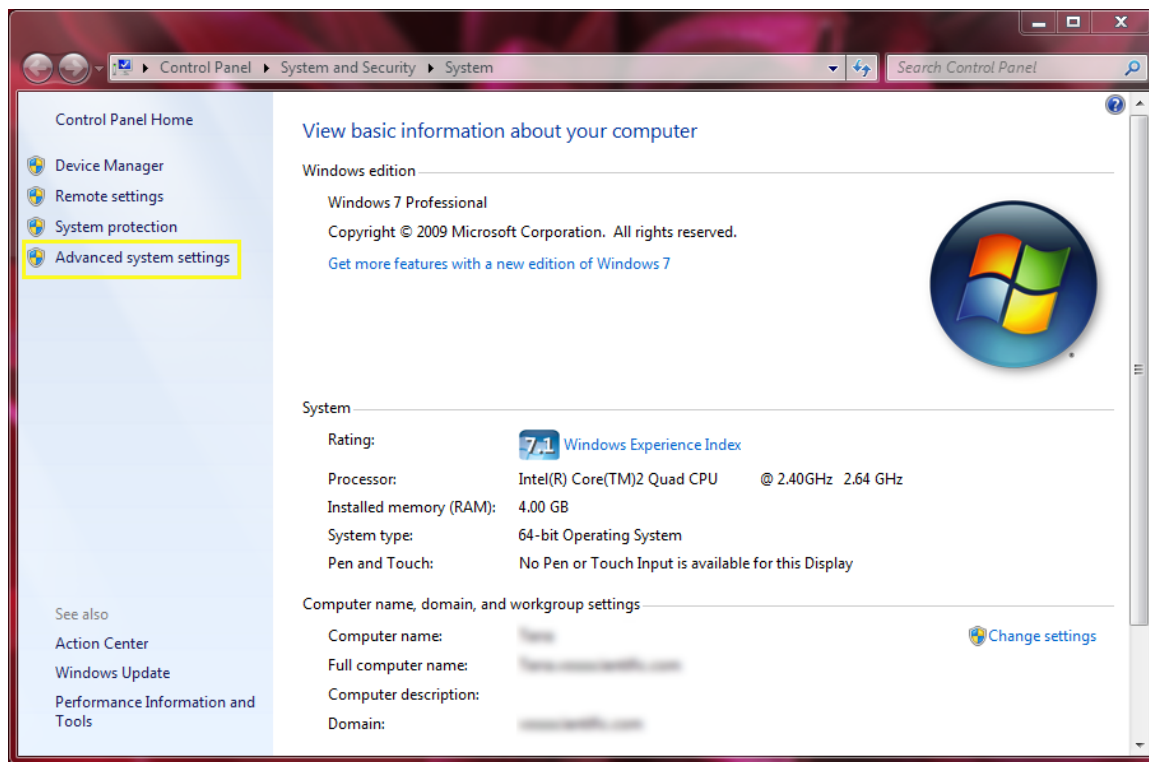
Add the mpich2 bin folder to the system path.

Right-click: Start Menu -> Computer

Select: Properties

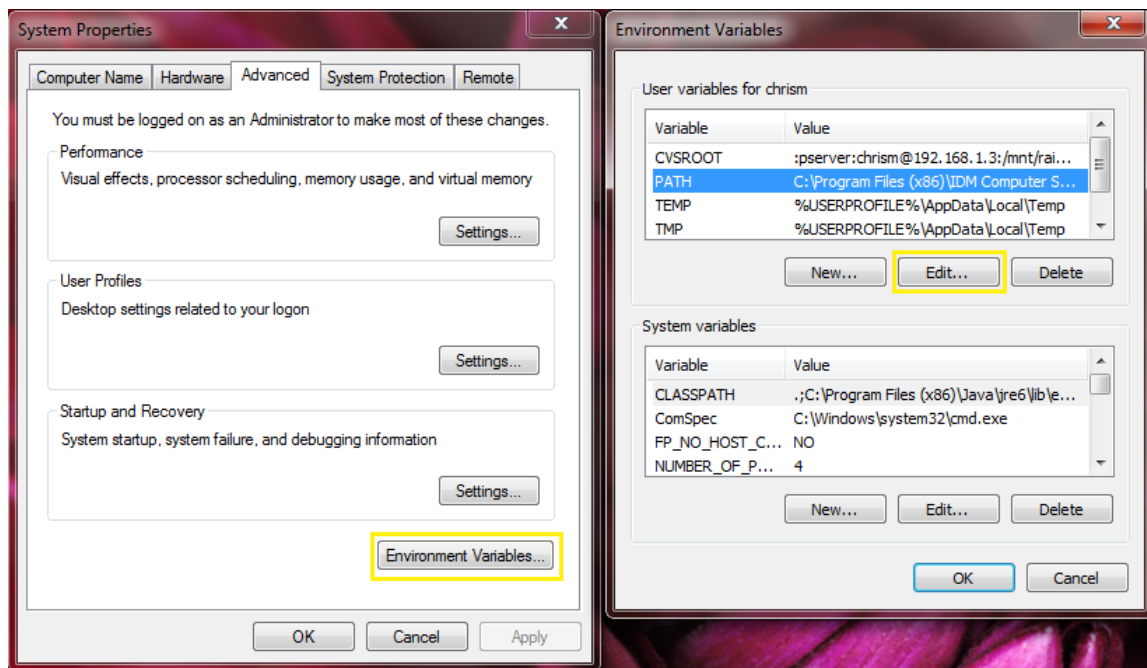


Select: Advanced System Settings

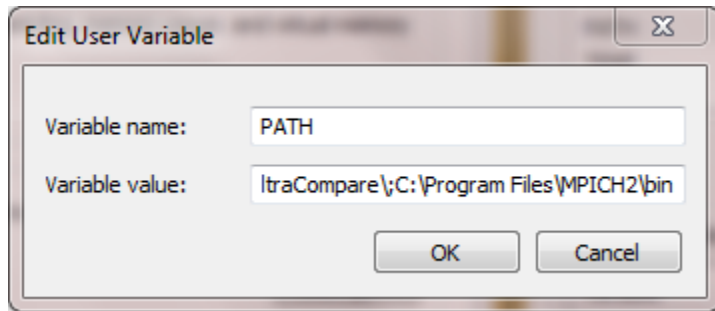


Click the Advanced tab and click Environment Variables.

Click the Path variable and click Edit. If Path does not exist click New.



Add the MPICH2 path at the end of the entry.



Run a command prompt as administrator as detailed previously.

Run: `smpd.exe -install`

## Test the MPI installation

A simple multi-process 64bit executable is available at:

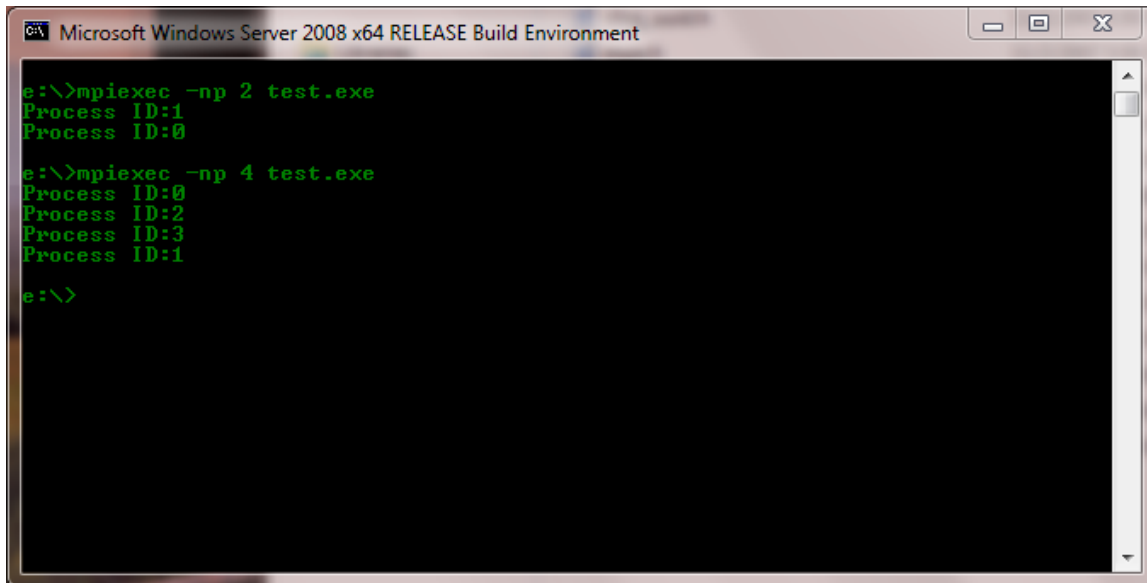
[http://www.vosssci.com/upload/ATK\\_Lsp/Common/VS-MP/test.exe](http://www.vosssci.com/upload/ATK_Lsp/Common/VS-MP/test.exe)

From a command prompt, test the mpi installation by running:

```
mpiexec -np 2 test.exe
```

(The initial command may ask permission for the mpi manager to access resources. It may also ask for an account and password. The account is the win domain account that you use to log on, and the password is the one that you log on with.)

The output should look as such:



```
Microsoft Windows Server 2008 x64 RELEASE Build Environment

e:\>mpiexec -np 2 test.exe
Process ID:1
Process ID:0

e:\>mpiexec -np 4 test.exe
Process ID:0
Process ID:2
Process ID:3
Process ID:1

e:\>
```

N.B., After computer has been turned off, the MPICH2 software may need to be reinstalled:

Run cmd.exe as an administrator:

CD to c:\program files\MPICH2\bin\

RUN: smpd.exe -install

To test CD to c:\lpsuite\multiprocess\ (if this is where test.exe is) and

RUN: mpiexec.exe -np 16 test.exe

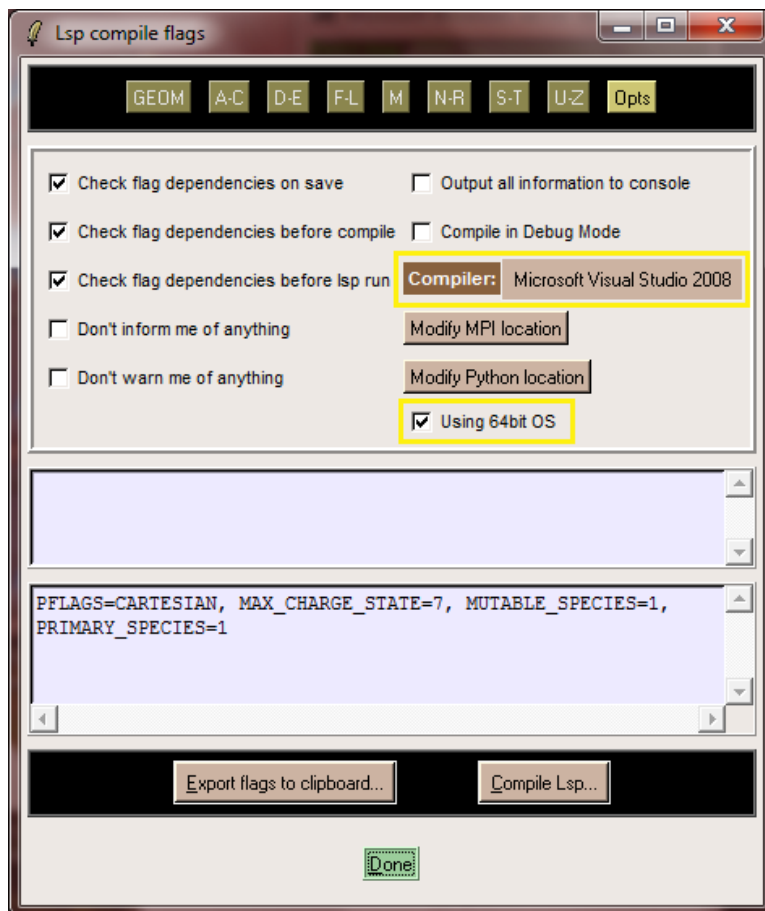
## Install and configure LspSuite

Install LspSuite in the usual manner.

Start GLsp and choose: Tools -> Lsp -> Compile

Click the Opts button at the top.

Click the compiler drop-down and select 'Microsoft Visual Studio 2008'



The MPI and Python locations will be chosen on the first compilation; however, they may be modified later.

All settings will be saved as the default.

Note: The 'MULTI\_PROCESS' and 'USE\_PYTHON' compiler flags must be enabled to compile with mpi and python.

## Running Multi-Process Inputs

From GLsp, choose: Tools -> Lsp -> Run

Set the 'Lsp preopt' entry as indicated below:

